COMP3630/6360: Theory of Computation
Semester 1, 2022
The Australian National University

Time Complexity

This lecture covers Chapter 10 of HMU: Time Complexity

- NP-Hardness of CNFSAT
- NP-Hardness of 3SAT
- More NP-Hard Problems

Additional Reading: Chapter 10 of HMU.

Cook's Theorem (*SAT* is **NP**-Complete)

- Cook's theorem gives a "generic reduction" for every problem in **NP** to *SAT*. So SAT is as hard as any other problem in **NP**—it's **NP**-complete.
- So, *SAT* is the granddaddy of all **NP**-complete problems.
- Many people have worked on the *SAT* problem, and there are now very efficient solvers (SAT solvers) for it.
- People frequently translate **NP**-complete problems to propositional logic, and then attack them with these general solvers!

CSAT

CSAT is a special case of *SAT*.

$$CSAT = \{\ \langle\phi\rangle \mid \phi \text{ is a satisfiable cnf formula } \}$$

where a Boolean formula is in *cnf* (for *conjunctive normal form*) if it is (also) generated by the grammar

$$\phi \to (c) \mid (c) \wedge \phi \qquad\qquad c \to \ell \mid \ell \vee c$$

$$\ell \to p \mid \neg p \qquad\qquad p \to x \mid y \mid \ \ldots$$

We call cs *clauses*, ℓs *literals*, and ps *propositions*.

Example 10.1

$(x \vee z) \wedge (y \vee z)$ is a cnf for the Boolean formula $(x \wedge y) \vee z$.

CSAT is **NP**-Complete

Clearly *CSAT* is in **NP** because we can use the same certificate for ϕ in cnf as we would for the same ϕ in *SAT*.

Giving a **P** reduction from *SAT* to *CSAT* is tricky.

A straight-forward translation of Boolean formulae into equivalent cnf may result in an exponential blow-up, meaning that this approach is useless.
Instead, we recall a reduction f won't have to preserve *satisfaction*:

$$\forall\pi\,(\pi \models \phi \quad \Leftrightarrow \quad \pi \models f(\phi))$$

but merely *satisfiability*

$$\exists\pi\,(\pi \models \phi) \quad \Leftrightarrow \quad \exists\pi\,(\pi \models f(\phi))$$

meaning that we're free to choose different πs for the two sides.

CSAT is **NP**-Hard

The translation from Boolean formulae to cnf proceeds in two steps which are both in **P**.

1. Translate to *nnf* (*negation normal form*) by pushing all negation symbols down to propositions. (This is still satisfaction-preserving.)
2. Translate from nnf to cnf. (This merely preserves satisfiability.)

Pushing Down $\neg$

We use de Morgan's laws and the law of double negation to rewrite left-hand-sides to right-hand-sides:

$$
\begin{aligned}
\neg(\phi \wedge \psi) &\Leftrightarrow \neg(\phi) \vee \neg(\psi) \\
\neg(\phi \vee \psi) &\Leftrightarrow \neg(\phi) \wedge \neg(\psi) \\
\neg(\neg(\phi)) &\Leftrightarrow \phi
\end{aligned}
$$

Example 10.2

$$
\begin{aligned}
\neg((\neg(x \vee y)) \wedge (\neg x \vee y)) &\Leftrightarrow \neg(\neg(x \vee y)) \vee \neg(\neg x \vee y) \\
&\Leftrightarrow x \vee y \vee \neg(\neg x \vee y) \\
&\Leftrightarrow x \vee y \vee \neg(\neg x) \wedge \neg y \\
&\Leftrightarrow x \vee y \vee x \wedge \neg y
\end{aligned}
$$

Pushing Down $\neg$ cont.

Theorem 10.3

Every Boolean formula ϕ is equivalent to a Boolean formula ψ in nnf. Moreover, $|\psi|$ is linear in $|\phi|$ and ψ can be constructed from ϕ in **P**.

Proof.

by induction on the number n of Boolean operators ($\wedge$, $\vee$, $\neg$) in ϕ we may show that there is an equivalent ψ in nnf with at most $2n - 1$ operators. $\square$

nnf $\longrightarrow$ cnf

Theorem 10.4

There is a constant c such that every nnf ϕ has a cnf ψ such that:

1. ψ *consists of at most* $|\phi|$ *clauses.*
2. ψ *is constructable from* ϕ *in time at most* $c|\phi|^2$.
3. $\pi \models \phi$ *iff there exists an extension* π' *of* π *satisfying* $\pi' \models \psi$, *for all interpretations* π *of the propositions in* ϕ.

Proof.

by induction on $|\phi|$. $\square$

Example 10.5

Consider $x \wedge \neg y \vee \neg x \wedge (y \vee z)$. An equisatisfiable cnf is
$(u \vee x) \wedge (u \vee \neg y) \wedge (\neg u \vee \neg x) \wedge (\neg u \vee v \vee y) \wedge (\neg u \vee \neg v \vee z)$.

3SAT

3SAT is a special case of *CSAT*.

$$3SAT = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3cnf formula } \}$$

where a Boolean formula is in *3cnf* (for *3 literal conjunctive normal form*) if it is (also) generated by the grammar

$$\phi \to (c) \mid (c) \wedge \phi \qquad\qquad c \to \ell \vee \ell \vee \ell$$
$$\ell \to p \mid \neg p \qquad\qquad p \to x \mid y \mid \ldots$$

Example 10.6

$(x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (x \vee \neg y \vee \neg z)$ is a 3cnf for the Boolean formula x.

3SAT is **NP**-Complete

Proof.

Clearly *3SAT* is in **NP** because we can use the same certificate for ϕ in 3cnf as we would for the same ϕ in *SAT* (or *CSAT*).

Sipser prefers to adapt his **NP**-hardness proof for *SAT* to *3SAT* over giving a **P** reduction from *SAT* to *3SAT*.

We **P** reduce from *CSAT* to *3SAT* instead, by translating arbitrary clauses into clauses with exactly three literals. $\square$

Proof detail: how to transform a cnf $\phi = \bigwedge_{i=1}^{n} c_i$ into an equisatisfiable 3cnf. We transform each clause $c_i = \bigvee_{j=1}^{k_i} \ell_{i,j}$ depending on the number k_i of literals in it. (We omit subscript i.)

Case $k = 1$ (ℓ_1) is replaced by

$$(\ell_1 \vee u \vee v) \wedge (\ell_1 \vee u \vee \neg v) \wedge (\ell_1 \vee \neg u \vee v) \wedge (\ell_1 \vee \neg u \vee \neg v)$$

for some fresh propositions u, v.

Case $k = 2$ $(\ell_1 \vee \ell_2)$ is replaced by

$$(\ell_1 \vee \ell_2 \vee u) \wedge (\ell_1 \vee \ell_2 \vee \neg u)$$

for some fresh proposition u.

Case $k = 3$ is 3cnf already.

Case $k > 3$ $(\bigvee_{j=1}^{k} \ell_j)$ is replaced by

$$(\ell_1 \vee \ell_2 \vee u_1) \wedge \bigwedge_{j=1}^{k-4} (\ell_{j+2} \vee \neg u_j \vee u_{j+1}) \wedge (\neg u_{k-3} \vee \ell_{k-1} \vee \ell_k)$$

for some $k - 3$ fresh propositions $u_1, \ldots, u_{k-3}$.

CLIQUE is **NP**-Complete

Let

$$CLIQUE = \left\{ \langle G, k \rangle \;\middle|\; \begin{array}{c} G \text{ is undirected graph} \\ \text{with } k\text{-clique} \end{array} \right\}$$

We show **NP**-completeness on the whiteboard.

HAMPATH is **NP**-Complete

Recall that

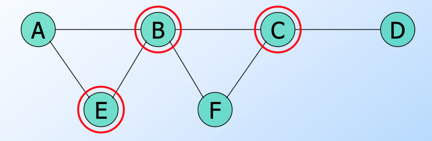
$$\mathit{HAMPATH} = \left\{ \langle G, s, t \rangle \;\middle|\; \begin{array}{l} \text{Directed graph } G \text{ has a} \\ \text{Hamiltonian path from } s \text{ to } t \end{array} \right\}$$

We already know that *HAMPATH* is in **NP**. We show **NP**-hardness by proving *3SAT* $\leq_{\mathbf{P}}$ *HAMPATH* on the whiteboard.

Node Cover

Given an undirected graph G, a *node cover* of G is a set C of vertices such that:

- for every edge between v_1 and v_2, one of v_1 or v_2 is in C.



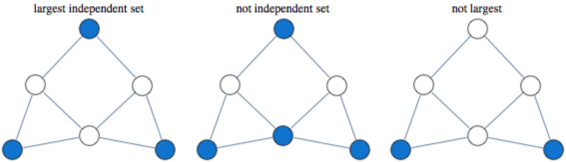
The *Node Cover Problem* is the problem of deciding whether a graph G has a node cover with k or fewer nodes:

$$NC = \{\langle G, k\rangle \mid G \text{ has node cover of size } \leq k\}$$

Independent Set

Given an undirected graph G, a *independent set* of G is a set C of vertices such that:

- no to vertices v_1 and $v_2 \in C$ are connected by an edge.

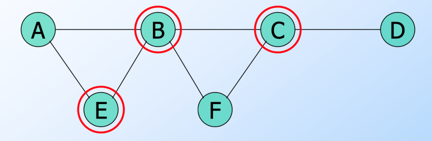


The *Independent Set Problem* is the problem of deciding whether a graph G has an independent set with k or or more nodes:

$$IS = \{\langle G, k\} \mid G \text{ has independent set of size } \geq k\}$$

Node Cover vs Independent Set

Q. How are node cover and independent set related?



Node Cover vs Independent Set II

Theorem. A graph G with n vertices has a node cover of size k iff it has an independent set of size $n - k$. Indeed, Node Cover is polytime reducible to independent set.

Corollary. If Node Cover is in NP, then so is independent set.

Theorem. Node Cover is in NP (whiteboard).